IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: Grieskamp et al.

FILED VIA EFS ON March 29, 2007

Application No. 10/650,245

Filed: August 27, 2003

Confirmation No. 7189

For: STATE AS A FIRST-CLASS CITIZEN OF AN IMPERATIVE LANGUAGE

Examiner: Thai Van Pham

Art Unit: 2192

Attorney Reference No. 3382-64897-01

COMMISSIONER FOR PATENTS
P.O. BOX 1450
ALEXANDRIA, VA 22313-1450

AMENDMENT

This responds to the Office action dated January 3, 2007. Please amend the referenced application as follows:

Amendments to the Claims are reflected in the listing of claims, which begins on page 2.

Remarks begin on page 7.

Amendments to the Claims

1. (Currently Amended) A computer readable medium comprising a data structure comprising computer executable instructions for a method for saving state for a semantically accessible state binding method, the ~~computer readable medium~~ method comprising:

defining a first state frame including a representation of a state of an executing program, comprising values of registers and heap associated with the executing program; and

defining a second state frame including a representation of state changes made by the executing program after the first state frame is created and the second state frame includes a pointer back to the first state frame.

2. (Currently Amended) The computer readable medium of claim 1 further comprising:

defining a third state frame including a representation of state changes made by the executing program after a fork method creates the third frame, and the third state frame includes a pointer back to the second frame.

3. (Currently Amended) The computer readable medium of claim 2 further comprising defining ~~wherein~~ a fourth frame which includes changes made by the executing program after the fork method creates the third state frame and after a set method returns the executing program to the state of the second state frame.

4. (Currently Amended) The computer readable medium of claim 3 further comprising defining a joined state frame including a combination of state changes in the third and fourth frames.

5. (Original) The computer readable medium of claim 3 wherein a first thread of the executing program makes state changes copied in the second frame, and a second thread of the executing program makes state changes copied into the third frame.

6. (Original) The computer readable medium of claim 1 wherein the second state frame includes unchanged state read from the first state frame.

7. (Currently Amended) A computerized method comprising:

receiving via an application programming interface a request to create a state save;

in response to the request, saving a first representation of a state of an executing program ~~in response to the request~~ comprising copying state of the program required to return to the moment the state was saved;

maintaining a second representation of subsequent state comprising changes made to the state of the executing program after the first representation; and

resetting the executing program to the saved first representation upon receiving a state set request at the application programming interface.

8. (Currently Amended) A computer system comprising:

memory and a central processing unit executing,

a program including executable instructions and an evolving present state; a state component comprising an initial representation of a prior evolving present state of the program, the initial representation comprising an instruction pointer location,

a subsequent representation of state changes made by the program since the initial representation, and

a method for returning the program state to the prior evolving present state using the instruction pointer location.

9. (Original) The computer system of claim 8 wherein the state component further includes a method for reading a location and value from the initial representation of the prior evolving present state and storing the value in a subsequent representation of the state changes.

10. (Original) The computer system of claim 8, wherein the state component includes a fork method for maintaining state for a thread spawned by the program and a forked representation of state changes made by the spawned thread of the program.

11. (Original) The system of claim 10 wherein the state component includes a join method for joining state changes made by the forked thread back into state changes of the subsequent representation.

12. (Canceled)

13. (Canceled)

14. (Canceled)

15. (Canceled)

16. (Canceled)

17. (Original) A computer readable medium comprising computer executable instructions for performing the method of claim 7.

18. (Canceled)

19. (Currently Amended) A computer readable method comprising computer executable instructions for performing a method comprising:

receiving a request from a method, which takes as a parameter a state object, to create a saved state of an executing model;

saving a first representation of a state of the executing model as the state object;

maintaining a second representation of state changes made by the executing model after the first representation; and

reinstating the executing model state to the state of the first representation using the state object.

20. (Canceled)

21. (Canceled)

22. (New) A computer-based process for allowing an executing version of a program to return to an earlier state, comprising:

defining a state component, the state component including:

a method for saving state of the program, the saved program state represented by a state token; and

a method for returning the program to the saved program state using the state token.

23. (New) The method of claim 22 wherein saving state of the program comprises saving heap and register values of the program at the time the state was saved.

24. (New) The method of claim 22 wherein saving state of the program comprises saving a program instruction pointer location at the time the state was saved.

25. (New) The method of claim 22 wherein at runtime the method for saving state of the program is invoked first and second times, and wherein saving state of the program the first time comprises saving heap and register values as a first frame; and wherein saving state of the program the second time comprises saving heap and register values whose values have changed since the saving state of the program the first time.

26. (New) The method of claim 25 wherein the method for returning the program to the saved program state comprises when a variable is accessed, checking the frames in opposite order to their creation until an update for the variable is found.

27. (New) A programming language construct operational to run on a computer system, the programs written in the programming language having an evolving state when executed, comprising:

A state class comprising:

A state object;

A fork method which saves state of the executing program as a first state when invoked with the state object;

A set method which sets the state of the executing program to the first state when invoked with the state object; and

A join method which joins the current state of the executing program with the first state when invoked with the state object.

28. (New) A computer program product comprising a storage medium having embedded thereon computer readable instructions for a process for restoring state of a variable, the process comprising:

Defining a first data structure which comprises a first saved state frame, the first saved state frame comprising heap and resource values necessary to save substantially all of the state of the program at a first given time;

Defining a second data structure which comprises a second saved state frame, the second saved state frame comprising the values that have changed between the first given time and a second given time;

Defining a state class that includes a method to restore state of the program to the second given time by accessing the first state frame and the second state frame.

29. (New) The computer program product of claim 29 wherein the method to restore state comprises following a pointer between the second and first state frames to discover a last updated instance of a variable.

Remarks

Applicant respectfully request reconsideration of the present application in view of the foregoing amendments and the following remarks. Claims 1-21 are pending in the application. Claims 1-21 are rejected. Claims 12-16, 18, and 20-21 have been canceled without prejudice with this amendment. No claims have been allowed. Claims 22-29 have been added. Claims 1, 7, 8, 19, 22, 27, and 28 are independent.

35 U.S.C. § 101 Rejections of Claims 1-6

The Examiner rejects claims 1-6 on the grounds that the claim recites a data structure that doesn't impart functionality, and does not define any acts being performed by the associated data structure, thereby failing to provide a tangible and concrete result. Applicant respectfully disagrees with the Examiner's characterization of the claims and relevant law, and believes that the claims in their previous state satisfied 35 U.S.C. § 101. Nevertheless, Applicant has amended the above-rejected claims in an effort to expedite prosecution. Specifically, the computer readable data structure has been recast as "A computer readable medium comprising a data structure comprising computer executable instructions for a method."

MPEP 2106 divides descriptive material into "functional descriptive material" – "data structures and computer programs which impart functionality when employed as a computer component" and "non-functional descriptive material" – music, literary works, and mere arrangements of data.

Functional descriptive material is statutory when stored on a computer-readable medium. *See Warmerdam*, 33 F.3d at 1360-61, 31 USPQ2d at 1759 (claim to computer having a specific data structure stored in memory held statutory product-by-process claim.) However, if just a data structure is claimed, without being stored on a computer-readable medium, then the claim is not statutory. *See Warmerdam*, 33 F.3d at 1361, 31 USPQ2d at 1760 (claim to a data structure *per se* not stored on a computer-readable medium held nonstatutory). Also see *In re Lowry*, 32 U.S.P.Q.2d 1031 ("memory" with a "data structure" held statutory.)

Claim 1, as amended, comprises function descriptive material – "a data structure comprising computer executable instructions," stored on a computer-readable medium. As such, applicants respectfully submit that Claim 1, as amended, is directed to statutory subject matter

and request that the rejection under 35 U.S.C. § 101 be withdrawn. Claims 2-6 depend on Claim 1 and at least for that reason should also not be subject to a 35 U.S.C. § 101 rejection.

35 U.S.C. § 102 Rejections of claims 7-9, 17, and 19-20

The Action rejects claims 7-9, 17, and 19 under 35 USC 102(b) as being anticipated by U.S. Patent Application No. 10/008,864 to Bates et al. (Bates). Applicants respectfully submit the claims are allowable over the cited art. For a 102(b) rejection to be proper, the cited art must show each and every element as set forth in a claim. (*See* MPEP § 2131.01.) However, the cited art does not describe each and every element. Accordingly, applicants request that all rejections be withdrawn.

Independent claim 7.

Amended independent claim 7 recites:

> A computerized method comprising:
> receiving via an application programming interface a request to create a state save;
> in response to the request, saving a first representation of a state of an executing program-comprising *copying values required to return to the moment the state was saved*;
> maintaining a second representation of subsequent state comprising changes made to the state of the executing program after the first representation; and
> resetting the executing program to the saved first representation upon receiving a state set request at the application programming interface.

Applicants respectfully assert that the Bates at a minimum, fails to teach or suggest "in response to the request, saving a first representation of a state of an executing program comprising *copying values required to return to the moment the state was saved*."

Bates teaches saving the state of specific variables which are being monitored. [Bates, Fig. 5, ¶46.] Each variable trigger value needs to be separately requested. That is, there is a one-to-one relationship between trigger values chosen and the variables whose state is saved. This neither teaches nor suggests the above-quoted claim language, where a request to create a save state leads to saving a first representation of a state of an executing program-comprising *essentially all* program variable values. As Bates neither teaches nor suggests the claim limitation "at least in response to the request, saving a first representation of a state of an

executing program-comprising *copying values required to return to the moment the state was saved*," applicants respectfully submit that claim 7 is in condition for allowance.

Dependent Claim 17

Claim 17 depends on claim 7. Thus, at least for the reasons set forth above with respect to claim 7, claim 17 should also be in condition for allowance. This claim also set forth independently patentable combinations.

Independent claim 8.

Amended independent claim 8 recites:

> A computer system comprising:
> memory and a central processing unit executing,
> a program including executable instructions and an evolving present state; a state component comprising an initial representation of a prior evolving present state of the program, the initial representation *comprising an instruction pointer location*,
> a subsequent representation of state changes made by the program since the initial representation, and
> a method for returning the program state to the prior evolving present state using the instruction pointer location.

Support for this amendment can be found in the specification, for example, at page 7, lines 11 to 18.

Applicants respectfully assert that the Bates at a minimum, fails to teach or suggest "a state component comprising an initial representation of a prior evolving present state of the program, the initial representation comprising an instruction pointer location." Bates teaches saving the state of various variables/expressions defined by a user. The states of the defined variable/expressions can be restored and stored in a history table. [Bates, ¶54.] This does not teach or suggest the above-quoted claim language. Rather, Bates teaches against such a limitation as only defined variables/expressions have their state saved and then restored. Thus, applicants believe that claim 8 is in condition for allowance.

Dependent Claim 9

Claim 9 ultimately depend on claim 8. Thus, at least for the reasons set forth above with respect to claim 8, claim 9 should also be in condition for allowance. This claim also sets forth

independently patentable combinations.

Independent claim 19.

Amended independent claim 19 recites:

> A computer readable method comprising computer executable instructions for performing a method comprising:
>
> receiving a request *from a method, which takes as a parameter a state object,* to create a saved state of an executing model;
>
> saving a first representation of a state of the executing model as the state object;
>
> maintaining a second representation of state changes made by the executing model after the first representation; and
>
> reinstating the executing model state to the state of the first representation *using the state object.*

Support for this amendment can be found in the specification, for example, at page 7, lines 11 to 18.

Applicants respectfully assert that the Bates at a minimum, fails to teach or suggest "receiving a request *from a method, which takes as a parameter a state object*, to create a saved state of an executing model." Bates does not discuss state objects. Bates, therefore, also does not discuss methods which take state objects as parameters. Thus, applicants respectfully submit that claim 19 is in condition for allowance.

Dependent Claim 20

Claim 20 depends on claim 19. Thus, at least for the reasons set forth above with respect to claim 19, claim 20 should also be in condition for allowance. This claim also set forth independently patentable combinations of method acts.

35 U.S.C. § 103 Rejections of claims 1-6

The Action rejects claims 1 and 6 under 35 U.S.C. 103(a) as being unpatentable over Bates. The action rejects claims 2-5, 10, and 11 under 35 U.S.C. 103(a) as being unpatentable over Bates in further view of U.S. Patent No. 6,463,527 to Vishkin (Vishkin). To establish a prima facie case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally

available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. (MPEP § 2142.)

Independent Claim 1

Amended Claim 1 recites:

> A computer readable medium comprising computer executable instructions for a method for saving state for a semantically accessible state binding method, the method comprising:
>
> defining a first state frame including a representation of a state of an executing program, comprising values of registers and heap associated with the executing program; and
>
> defining a second state frame including a representation of state changes made by the executing program after the first state frame is created and the second state frame includes a pointer back to the first state frame.

Applicants respectfully assert that the Bates at a minimum, fails to teach or suggest "defining a first state frame including a representation of a state of an executing program, *comprising values of registers and heap associated with the executing program.*" Applicant has modified claim 1, support for the modification can be found at page 7, lines 11-21.

The Specification defines a representation of a state of an executing program, e.g., as follows:

> Here, conceptually with state as a first class citizen, the state of the program is copied by capture_state(). In one example, if the program includes a database, it is copied along with variables and other state required to return to the moment the state was saved. In one example, the saved state includes IP, CALL STACK, REGISTERS, HEAP, etc. The saved state can be modified, for example by setting the instruction pointer (IP) to a different address, so that when the saved state is restored the program execution will continue at a different point. In another example, only the REGISTERS and the HEAP are saved.
>
> So without knowing the details of the state of the program, the component user may save or recall all the potential variables at that moment in time, since a request to save the state of the program saves all this information. [Specification, page 7, lines 11-21.]

Bates saves individual variable values taken at specific trigger times. [Bates, ¶10.] Attached expressions may also be saved with the variables. [Bates, ¶10.] Saving a handful of

variables at a handful of times does not each or suggest "defining a first state frame including a representation of a state of an existing program, *comprising values of registers and heap associated with the executing program.*" Bates does not discuss register and heap values. Vishkin (cited against dependent claims 2-5) also does not discuss register and heap values. Thus, applicants respectfully submit that claim 1 is in condition for allowance.

Dependent Claims 2-5

The action rejects claims 2-5 under 35 U.S.C. 103(a) as being obvious over Bates in further view of Vishkin. Applicants respectfully assert that claims 2-5 recite novel and nonobvious features allowable over a Bates-Viskin combination. Further, since claims 2-5 depend from allowable claim 1, they should be allowed for at least those reasons stated for claim 1. Claims 2-5 should be allowable. Such action is respectfully requested.

Dependent Claim 6

Claims 6 ultimately depend on claim 1. Thus, at least for the reasons set forth above with respect to claim 1, claim 6 should also be in condition for allowance. This claim also set forth independently patentable combinations.

Dependent Claims 10-11

The action rejects claims 10 and 11 under 35 U.S.C. 103(a) as being obvious over Bates in further view of Vishkin. Applicants respectfully assert that claims 10-11 recite novel and nonobvious features allowable over a Bates-Viskin combination. Further, since claims 10-11 depend from allowable claim 8, they should be allowed for at least those reasons stated for claim 81. Claims 10-11 should be allowable. Such action is respectfully requested.

New Claims

Support for the new claims can be found in the application at, e.g., pages 4, 5, 7, and 19.

Formal Request For Interview

Upon reviewing this response, if any issues remain, the Examiner is formally requested to contact the undersigned prior to issuance of the next Office Action in order to arrange a

telephonic interview. It is believed that a brief discussion of the merits of the present application may expedite prosecution. Applicants submit the foregoing formal Response so that the Examiner may fully evaluate Applicants' position, thereby enabling the interview to be more focused. This request is being submitted under MPEP § 713.01, which indicates that an interview may be arranged in advance by a written request.

Conclusion

Claims 1-11, 17, 19, and 22-29 should be allowable. Such action is respectfully requested.

Respectfully submitted,

KLARQUIST SPARKMAN, LLP

One World Trade Center, Suite 1600
121 S.W. Salmon Street
Portland, Oregon 97204
Telephone: (503) 595-5300
Facsimile: (503) 595-5301

By /Genie Lyons/
Genie Lyons
Registration No. 43,841